Keep any notes you want here -- we can be more verbose than in our log with Victoria

mar 10

\*note : eventually add address calculation to ID

ADD:

IF

taken <- 0 (Default -- how do we make sure this is default? -- maybe and with signal from opcode)

adapt taken to account for JMP and JSR (eventually -- will be uncond for JMP/JSR -- watch timing)

fetch instr

rip/split in register between IF/ID

ID

selA = 0 (ctrl unit)

destSel determined by whatever is in WB (and R7 save eventually for JSR)

\*need to study why/how to do that half & half clock cycle register rw

input, regwrite -- irrelevant, set in WB

Ex

ALUSel1 = 0

ALUop set by control word

immsel passed through as part of instru to choose reg or imm

then ALUSel2 = 0

OffSel = X

PCOffSel = X

Mem  
NADA just pass through ALUout

write, read signals set possibly in control word but should be 1

AddrSel = X

Don’t wait for resp to load next reg -- load could be a logical combo of whether w/r and resp

WB

MemArithSel - signal that indicates arith or mem -- should indicate arith

ld\_Reg is a ctrl signal, should be 1

set destSel to 0

setCC -- should be set as a control signal

AND is the same as ADD except with a different ALUop control signal

LDR

fetch same

ID

selA = 0

Ex

use AExt6, so offsel = 00

alusel1 = 0

alusel2 = 1

aluop = ADD

pcoffsel = X

Mem

AddrSel = 0 for now, but will need logic for LDI

\*\*\*NOTE FOR FUTURE LDI: ADD FLIPFLOP FOR TRACKING LD STATE

Read = 0, writes are 1 << CONTROL sigs

wait for resp before loading to bar -- watch how we store MDR for sake of LDI

WB

memarithsel = 1 (doing mem)

ld\_Reg =1

DR passed

destSel = 0

setCC =1

STR

only what’s diff from LDR:

ID

same

Ex

SR is passed through

Now we want write lows & read high

input is automatically SR

WB

setCC = 0

ld\_Reg = 0

destSel = X

MemArithSel = x

set something so nothing is done during NOP -- for mem, reset state? (make own instr)

NOP: is BR 00000000000000000000000000

ID controls: 1 b

Sel A

Execute controls: 8 b

OffSel,

PCOffSel,

ALUSel1, ALUSel2

ALUop

Memory: 5 b

AddrSel

BR

mwriteH\_L

mwriteL\_L

mread\_L

WB: 4 b

setCC

memArithSel

ld\_Reg

DestSel

ROM:

18 bits ( 15 signals/vectors, but OffSel is 2b and ALUop is 3 )

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ADD | AND | LDR | STR | BR |
| selA | 0 | 0 | 0 | 0 | X |
| OffSel | XX | xx | 00 (Aoff6) | 00 | XX |
| PCOffSel | X | x | x | x | 0 (PCoff9) |
| ALUSel1 | 0 | 0 | 0 | 0 | 1 |
| ALUSel2 | 0 | 0 | 1 | 1 | X |
| ALUop | ADD | AND | ADD | ADD | PASS |
| AddrSel | X | x | 0 | 0 | X |
| BR | 0 | 0 | 0 | 0 | 1 |
| setCC | 1 | 1 | 1 | 0 | 0 |
| memArSel | 0 (Arith) | 0 | 1 (mem) | 1 | X |
| ld\_Reg | 1 | 1 | 1 | 0 | 0 |
| DestSel | 0 | 0 | 0 | X | X |
| mwriteH\_L | 1 | 1 | 1 | 0 | 1 |
| mwriteL\_L | 1 | 1 | 1 | 0 | 1 |
| mread\_L | 1 | 1 | 0 | 1 | 1 |

OP\_ADD

selA := ‘0’;

OffSel := “00”;

PCOffSel := ‘0’;

ALUSel1 := ‘0’;

ALUSel2 := ‘0’;

ALUop := ALU\_ADD;

AddrSel := ‘0’;

BR := ‘0’;

setCC := ‘1’;

memArSel := ‘0’;

ld\_Reg := ‘1’;

DestSel := ‘0’;

mwriteH\_L := ‘1’;

mwriteL\_L := ‘1’;

mread\_L := ‘1’;

OP\_AND

selA := ‘0’;

OffSel := “00”;

PCOffSel := ‘0’;

ALUSel1 := ‘0’;

ALUSel2 := ‘0’;

ALUop := ALU\_AND;

AddrSel := ‘0’;

BR := ‘0’;

setCC := ‘1’;

memArSel := ‘0’;

ld\_Reg := ‘1’;

DestSel := ‘0’;

mwriteH\_L := ‘1’;

mwriteL\_L := ‘1’;

mread\_L := ‘1’;

OP\_LDR

selA := ‘0’;

OffSel := “00”;

PCOffSel := ‘0’;

ALUSel1 := ‘0’;

ALUSel2 := ‘1’;

ALUop := ALU\_ADD;

AddrSel := ‘0’;

BR := ‘0’;

setCC := ‘1’;

memArSel := ‘1’;

ld\_Reg := ‘1’;

DestSel := ‘0’;

mwriteH\_L := ‘1’;

mwriteL\_L := ‘1’;

mread\_L := ‘0’;

OP\_STR

selA := ‘0’;

OffSel := “00”;

PCOffSel := ‘0’;

ALUSel1 := ‘0’;

ALUSel2 := ‘1’;

ALUop := ALU\_ADD;

AddrSel := ‘0’;

BR := ‘0’;

setCC := ‘0’;

memArSel := ‘1’;

ld\_Reg := ‘0’;

DestSel := ‘0’;

mwriteH\_L := ‘0’;

mwriteL\_L := ‘0’;

mread\_L := ‘1’;

OP\_BR

selA := ‘0’;

OffSel := “00”;

PCOffSel := ‘0’;

ALUSel1 := ‘1’;

ALUSel2 := ‘0’;

ALUop := ALU\_PASS;

AddrSel := ‘0’;

BR := ‘1’;

setCC := ‘0’;

memArSel := ‘0’;

ld\_Reg := ‘0’;

DestSel := ‘0’;

mwriteH\_L := ‘1’;

mwriteL\_L := ‘1’;

mread\_L := ‘1’;

CTRL\_WD <= selA & OffSel & PCOffSel & ALUSel1 & ALUSel2 &ALUop & AddrSel & BR & setCC &memArSel & ld\_Reg & DestSel & mwriteH\_L & mwriteL\_L & mread\_L;